

Display photos randomly with PHP

Do you have loads of pictures that you are keen to show off? Matt Preston explains how to view snaps at random with PHP

PHP is a popular way of creating dynamic webpages. It runs on the web server, generating standard HTML pages that are sent to the visitor. Any web browser can view a PHP page. In this month's *Web Expert*, we'll show you how PHP can be used to display a random photograph from your collection. All the files we use are on the cover disc and can be downloaded from www.shopperexpert.co.uk, which also houses the working demonstrations.

PHP files are normal text files, so you can do all your PHP work using Notepad. However, these files must end in .php to work. These instructions worked exactly in our WebFusion account, but you may need to alter file locations slightly if you're using a different hosting account. You can view our sample at www.shopperexpert.co.uk/randomimage/test-page.php.

To try our test files on your own web server, you need to use the same directory structure as us. If you want a more detailed explanation of the PHP commands we've used, there's excellent documentation online at www.php.net/docs.php.

GETTING STARTED

Create a folder called 'scripts' in the server's root and upload the common.php file. Create a folder called 'randomimage' and upload the remaining files and images folder here. Open Notepad and test-page.php and you'll see a file that looks like normal HTML, except there's a block of code in the middle:

```
<?php
require "../scripts/common.php";
$randomImage = getRandomImagePath
("../images");
?>
<p><?php
echo "<img src=\"\$randomImage\">";
?></p>
```

The <?php codes start a new block of PHP code; the ?> codes end it. As you can see, there are two blocks of PHP code. The first block uses 'require' to include another PHP file stored in the scripts folder in the server's root. It's this file that contains the function

(reusable code that does a specific job) for showing the random image, and we'll show you how to create that later. You should save your PHP functions into separate files so you can reuse them in other scripts later.

The second line of the script does the real work. It uses the getRandomImagePath() function to search the directory we want ("./images", in this case) for a random picture. It stores the file's name in the \$randomImage variable. A variable is a virtual storage box for data, and anything starting with a \$ is a variable. The second block of code uses the 'echo' command to output an HTML img tag using the contents of our variable as the image source.

INSIDE THE COMMON.PHP FILE

The real work is done inside the common.php file. If you open it, you'll see that it contains many functions. These are all used to display a random function. The first one is the getRandomImagePath that we used in our test-page.php file:

```
function getRandomImagePath($aDirectory)
{
    $imagePaths = getImagesInPath
        ($aDirectory);
    $size = count($imagePaths);

    if($size > 0)
    {
        $randomIndex = mt_rand(0, $size - 1);
        return $imagePaths[$randomIndex];
    }
    else
    {
        return false;
    }
}
```

This function stores the directory it is passed in the \$aDirectory variable. The next line calls a function called getImagesInPath with our directory to create an array (list) of all the images found. This is stored in the \$imagePaths variable. Arrays are indexed lists, where each entry can be accessed through its index number;

\$imagePaths[0] refers to the first item in the list, as arrays are indexed from 0, not 1.

Next, we use the built-in count() function to see how big the list is, and store its size in the \$size variable. We then check to see if \$size is greater than 0. If it isn't then the list is empty, and we return false. If it is then we need to pick a random image from our list. First, we store a random number in the \$randomIndex variable. This uses the built-in mt_rand() function, which creates a random image between its two inputs (0 is the first entry and \$size - 1 is the last, as arrays are indexed from 0). Finally, we return to the calling script the image stored at \$randomIndex using \$imagePaths[\$randomIndex].

INTRODUCTION

Welcome to the *Web Expert* column. Each month, we'll show you a new technique that you can use to create better websites. Each topic we cover will be a complete project that you can either use directly or modify as you need.

Many, but not all, of our projects require dedicated web hosting. It's a worthy investment if you want to create the best websites, as you'll get access to more features, have dedicated email and more web space than with any free accounts you have.

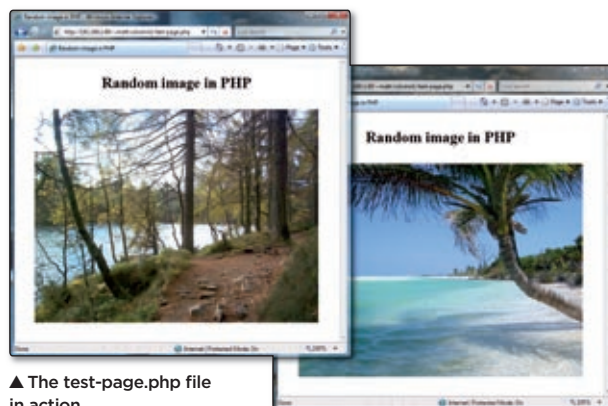
This month, we'll be showing you how to use the scripting language PHP to automatically display a photo randomly from a collection. You'll need to have dedicated hosting for this project. We're using WebFusion's Fusion Starter account (www.webfusion.co.uk), which costs £5.95 per month. Everything you read here is demonstrated at www.shopperexpert.co.uk.

Matt Preston

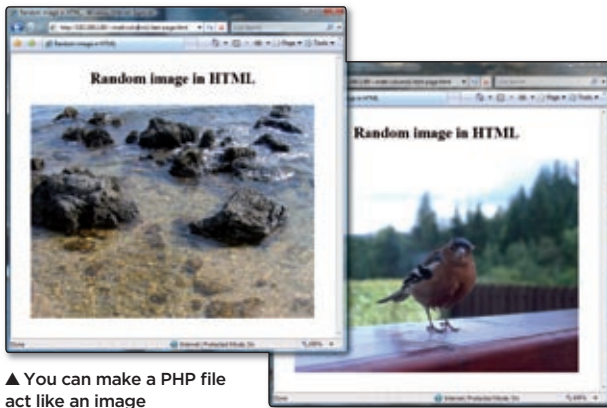
Web developer



matt@computersshopper.co.uk



▲ The test-page.php file in action



▲ You can make a PHP file act like an image

To create the list of images, the `getImageInPath` function looks like this:

```
function getImageInPath
($aDirectory)
{
    if ($handle = opendir($aDirectory))
    {
        while (($filename = readdir
            ($handle)) !== false)
        {
            $nextPath = "$aDirectory/
                $filename";

            if(isJpegFile($nextPath))
            {
                $imagePaths[] = $nextPath;
            }
        }

        closedir($handle);
    }
    return $imagePaths;
}
```

This function attempts to open the directory stored in the `$aDirectory` variable using the built-in `opendir()` function. If this directory can't be accessed, this returns false. Otherwise, it returns a 'handle', which we need to access the files. We store this in the `$handle` variable. Next, we use a while loop (while a statement is true, we perform every instruction inside the loop) to read through the directory and examine each file in turn. We use `readdir($handle)` to read a file and store the result in a variable called `$filename`. This function returns false when all files have been read, and this is our cue to cancel the loop.

For each file we read, we use the `$nextPath` variable to store the directory and filename of a file (`./image/picture.jpg`, for example).

Next, we see if the file stored in `$nextPath` is a JPEG using the `isJpegFile()` function. If it is, we store the result in an array called `$imagePaths` (the empty square brackets mean 'create a new list entry'). We use `closedir($handle)` to close our access to a file. When the while loop has ended, we return our list of

JPEG files. `IsJpegFile` uses two functions:

```
function isJpegFile($aPath)
{
    if(is_file($aPath))
    {
        return endsWith(strtolower
            ($aPath), ".jpg");
    }
    else
    {
        return false;
    }
}

function endsWith($aString, $aSuffix)
{
    $stringLength = strlen($aString);
    $suffixLength = strlen($aSuffix);

    if($stringLength < $suffixLength)
    {
        return false;
    }

    $endOfString = substr($aString,
        $stringLength - $suffixLength);
    return $endOfString == $aSuffix;
}
?>
```

The first function takes a path to a file. We check if there is a file, using `is_file()`. If there is, we want to know if the file ends with `.jpg` using the `endsWith()` function. We convert everything to lower case using the `strtolower()` function, to avoid complications from upper-case characters.

The `endsWith` function works out the length of the string and the length of the suffix we're

searching for. We want to separate the file's suffix and store it in the `$endOfString` variable. We do this using `substr` (sub-string), which takes two arguments: the string we want to cut up, and the numerical character position where we want to start our cut. This returns just the suffix. Next, we return `$endOfString == $aSuffix` (true if they match and false if they don't).

It's now easy to write a new function to look for other file types. If we wanted `isGifFile()`, we'd create it in the same way, but we'd use the line `return endsWith(strtolower($aPath), ".gif")`.

MAKING IT BETTER

A neater solution is to have a PHP script we can integrate into an HTML page. If you visit www.shopperexpert.co.uk/test-page.html, you'll see this in action. The code for the HTML page uses `{img src="random-image.php"}` For this to work, we make `random-image.php` act as an image file. If you view its code in Notepad, you'll see it uses the same code and `getRandomImagePath()` function as our first example.

The remaining code makes the PHP file act like an image file. `$handle = fopen($imagepath, 'rb')` opens the physical JPEG file. The 'header' code – `header("Cache-Control: no-store, no-cache, must revalidate", etc)` – is used to tell a web browser that there's a JPEG image coming, where it's stored, how big it is and, as we want a different image each time, not to store the file in its cache. The `fpassthrough()` line tells PHP to stream the contents of the file to the browser – in other words, send the JPEG image through.

We now have code we can use more than once in several webpages. You may want to modify this project to suit the way your website works. If you use different directories to us, update the path to the `common.php` file. You'll also need to point people towards the directory in which you keep your images.

We've shown you how this project can be extended to look for different image types such as GIFs, but what about making your script look for both JPEGs and GIFs? You could use an OR statement to see if a field is a JPEG file or GIF file. Read the PHP documentation for help. **ES**

Next month

BETTER WEB DESIGN

Find out how to use CSS to create beautiful curved boxes the easy way

NOBODY HOSTS MORE WEBSITES IN THE UK THAN US. FACT!

*Source: Netcraft

WEB HOSTING

FROM
£5.95
PER MONTH

VPS

FROM
£14.99
PER MONTH

DEDICATED SERVERS

FROM
£59.99
PER MONTH

RESELLER PACKAGES

FROM
£26.99
PER MONTH

WebFusion
WHERE WEBHOSTING WORKS